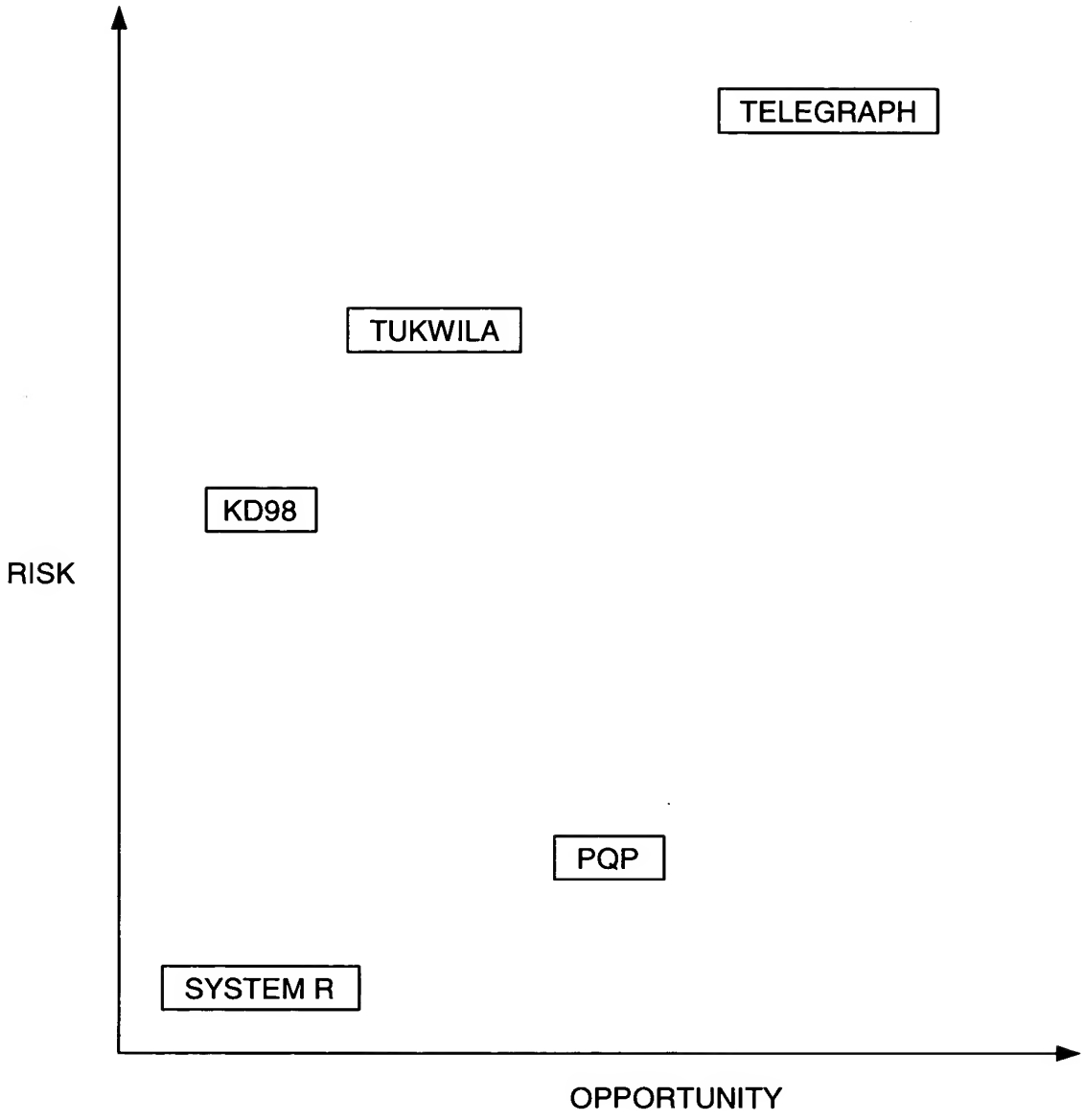
FIG. 1



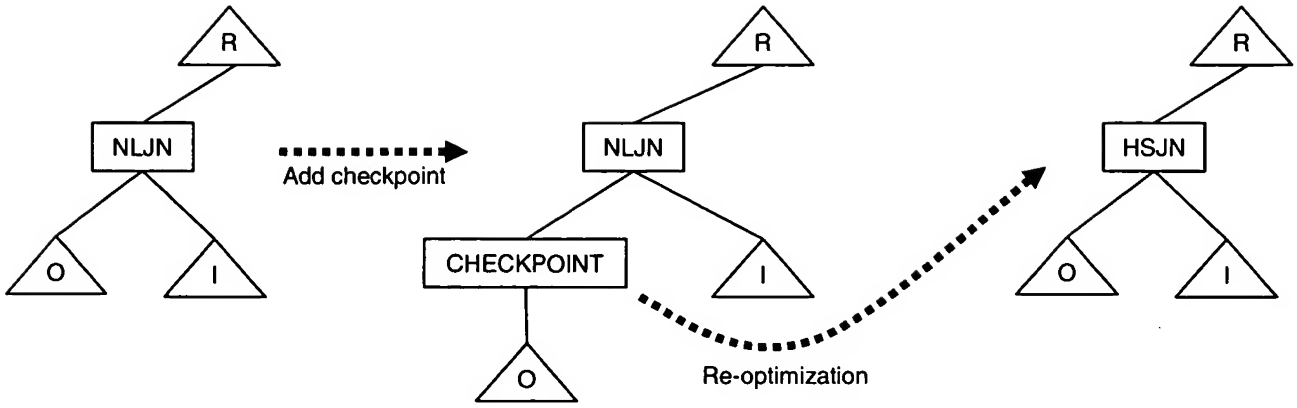

R
NLJN
O
I
Add checkpoint
R
NLJN
CHECKPOINT
O
I
Re-optimization
R
HSJN
O
I

FIG. 2

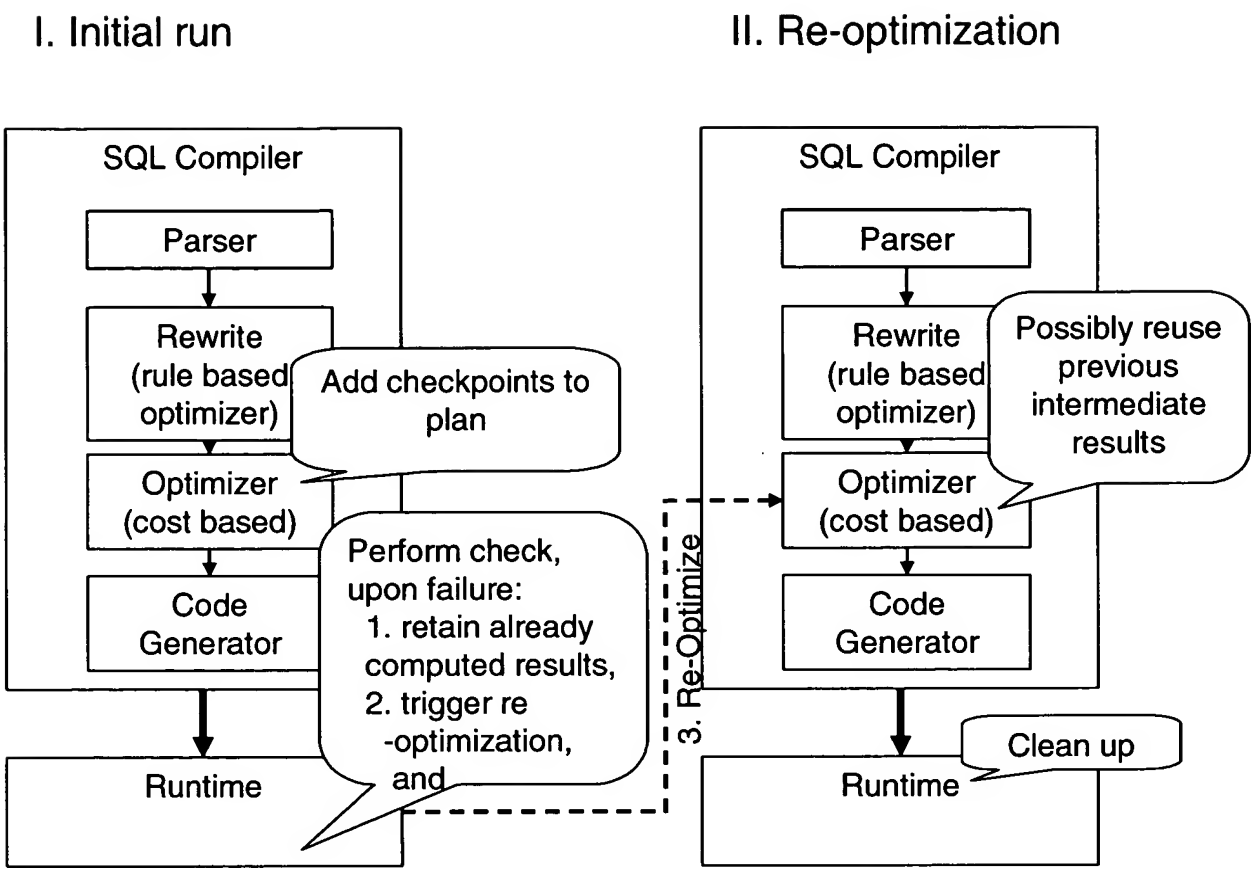

I. Initial run
II. Re-optimization
SQL Compiler
Parser
Rewrite (rule based optimizer)
Optimizer (cost based)
Code Generator
Runtime
Add checkpoints to plan
Perform check, upon failure:
1. retain already computed results,
2. trigger re -optimization, and
3. Re-Optimize
SQL Compiler
Parser
Rewrite (rule based optimizer)
Optimizer (cost based)
Code Generator
Runtime
Possibly reuse previous intermediate results
Clean up

FIG. 3

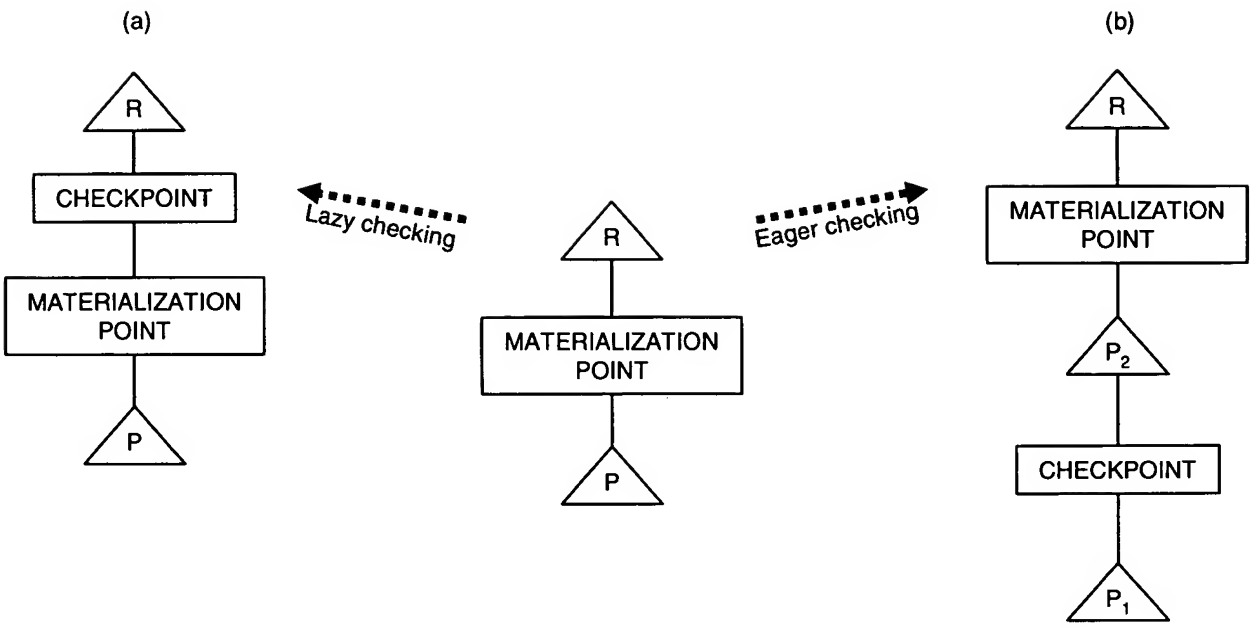

(a)
R
CHECKPOINT
MATERIALIZATION POINT
P
Lazy checking
R
MATERIALIZATION POINT
P
Eager checking
(b)
R
MATERIALIZATION POINT
P2
CHECKPOINT
P1

FIG. 4

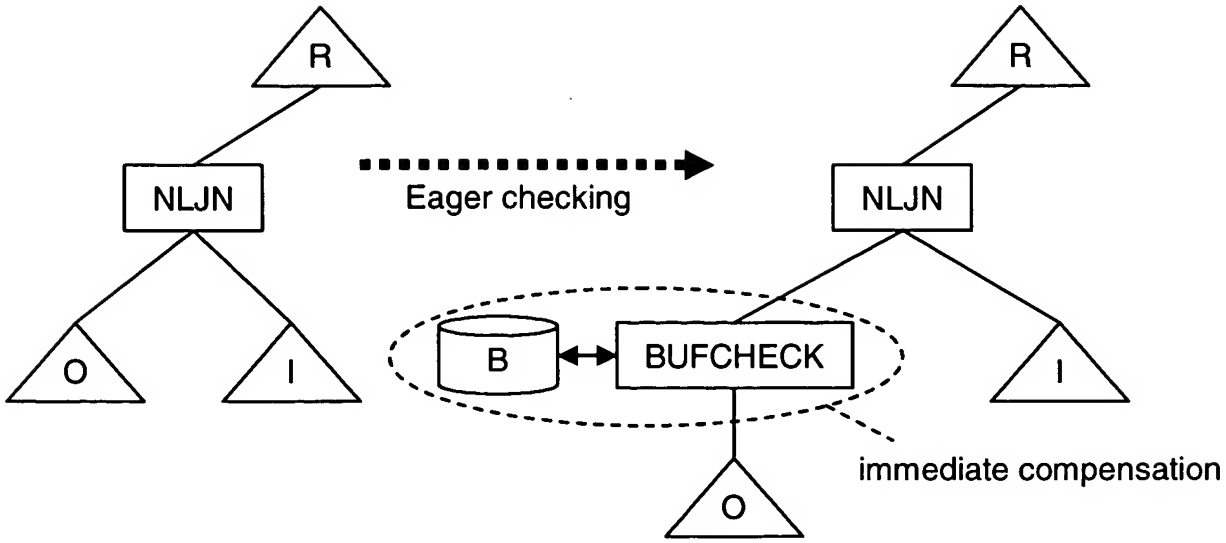

R
NLJN
O
I
Eager checking
R
NLJN
B
BUFCHECK
I
O
immediate compensation

FIG. 5

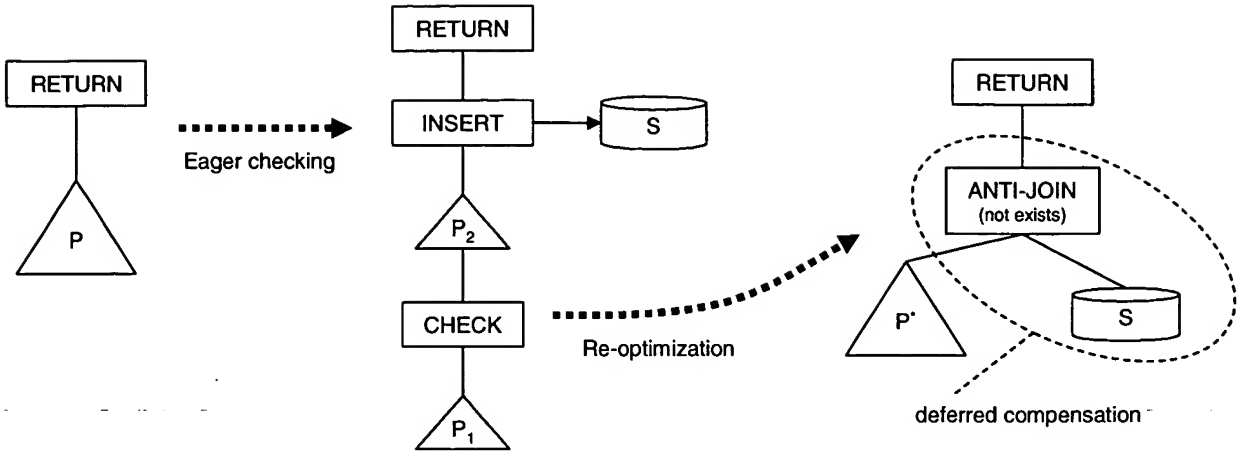

RETURN
P
Eager checking
RETURN
INSERT
S
P2
CHECK
P1
Re-optimization
RETURN
ANTI-JOIN
(not exists)
P'
S
deferred compensation

FIG. 6

```
CHECK.OPEN:
  count = 0;
CHECK.NEXT:
  count++;
  r = childStream.next();
  if count > high
    call re-optimization;
  if count < low and r = EOF
    call re-optimization;
  else
    return r;
CHECK.CLOSE:
  ∅
```

```
BUFCHECK.OPEN:
  count = 0;
  allocate B of size b;  // buffer
  for i = 0 to b do
    B[i] = childStream.next();
    if childStream.EOF()
      and i < low
    call re-optimization;
BUFCHECK.NEXT:
  count++;
  if high < count
    call re-optimization;
  if count < b
    return B[count];
  else
    return childStream.next();
BUFCHECK.CLOSE:
  free B;
```

FIG. 7

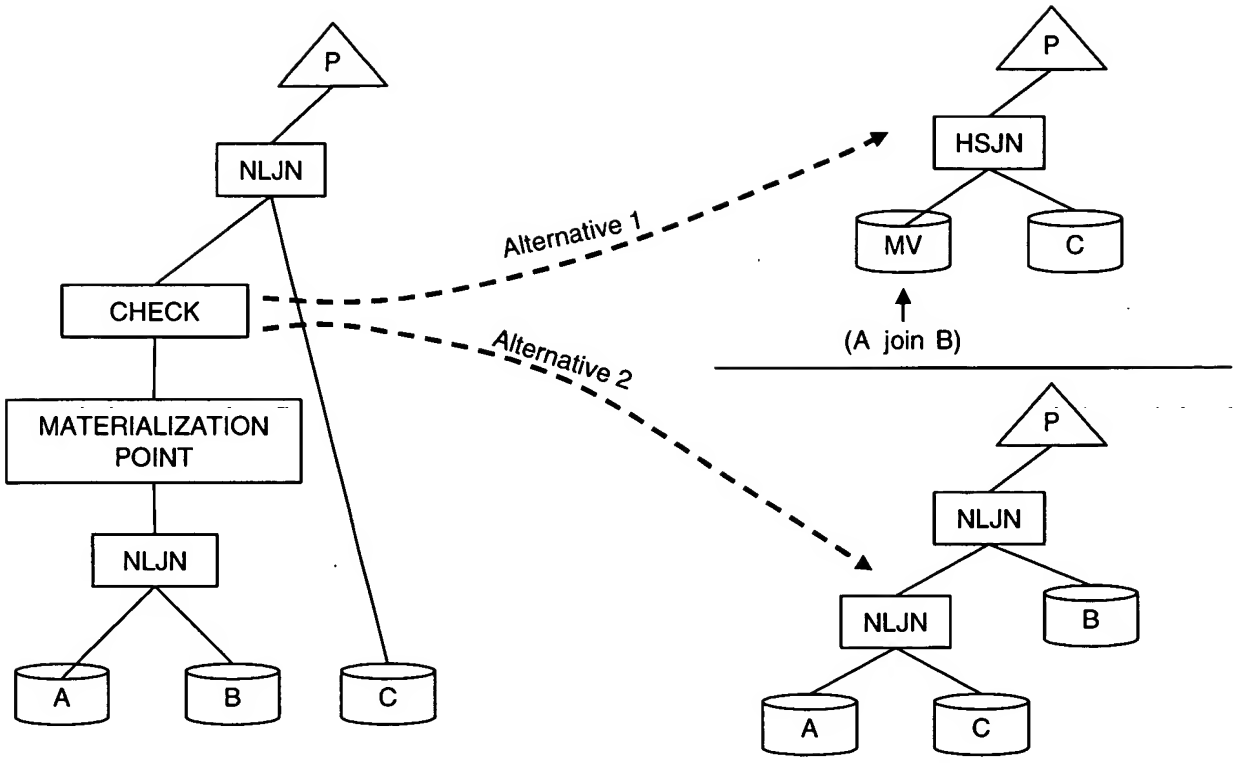

P
NLJN
CHECK
MATERIALIZATION POINT
NLJN
A
B
C
Alternative 1
Alternative 2
P
HSJN
MV
C
(A join B)
P
NLJN
NLJN
B
A
C

FIG. 8

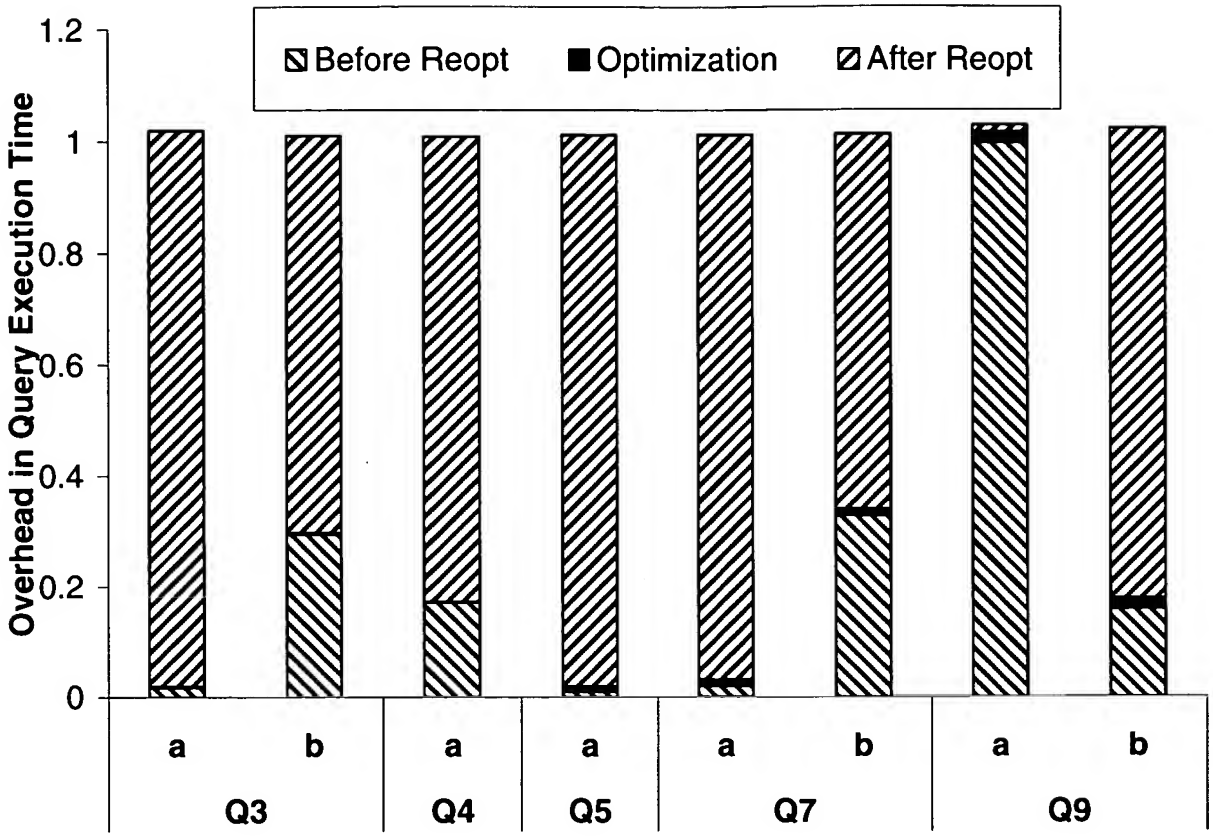

Before Reopt
Optimization
After Reopt
Overhead in Query Execution Time
1.2
1
0.8
0.6
0.4
0.2
0
a
b
a
a
a
b
a
b
Q3
Q4
Q5
Q7
Q9

FIG. 9

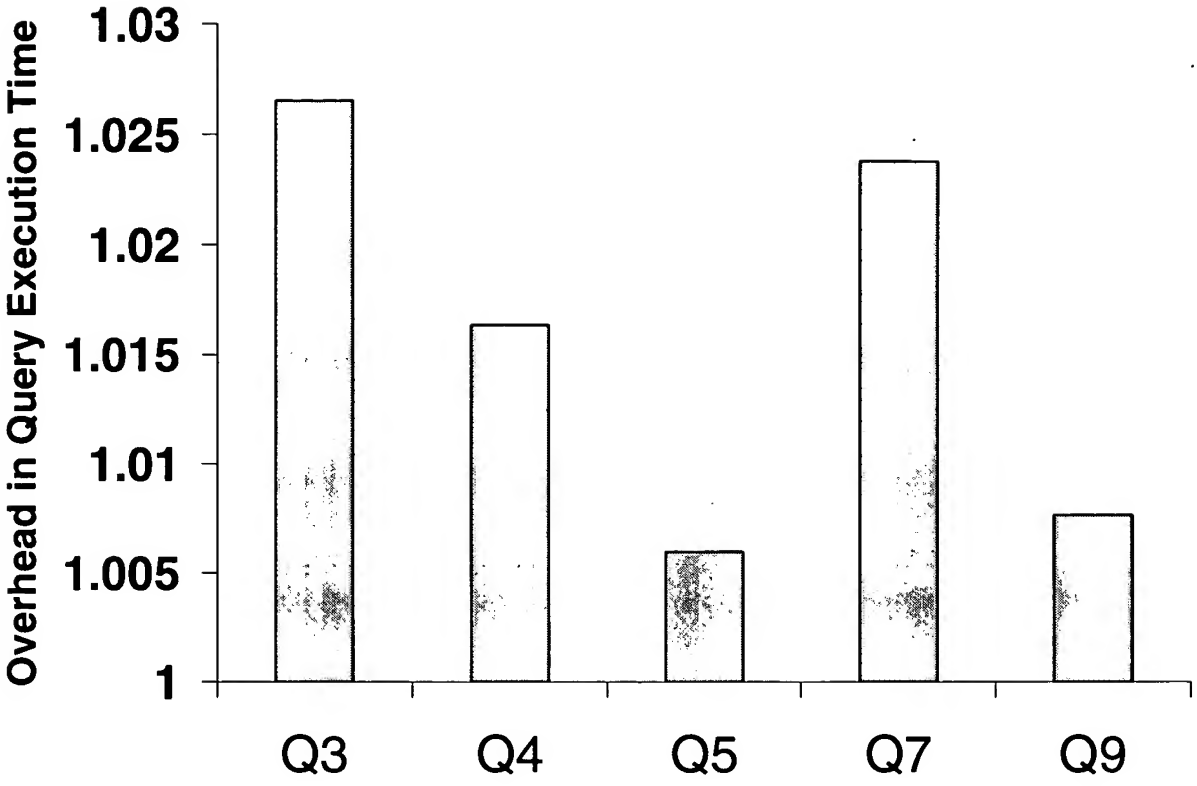

Overhead in Query Execution Time
1.03
1.025
1.02
1.015
1.01
1.005
1
Q3
Q4
Q5
Q7
Q9

FIG. 10

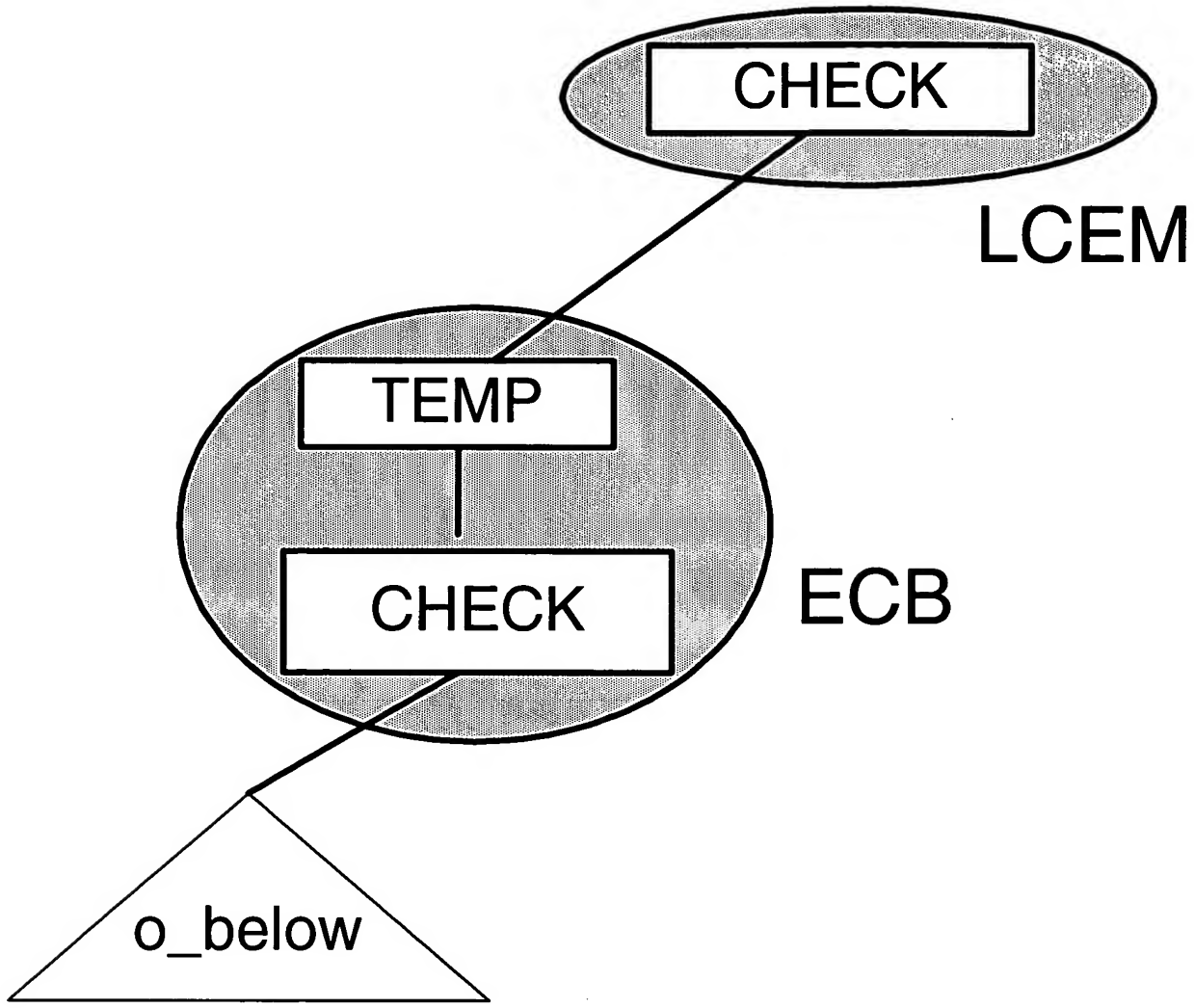


FIG. 11

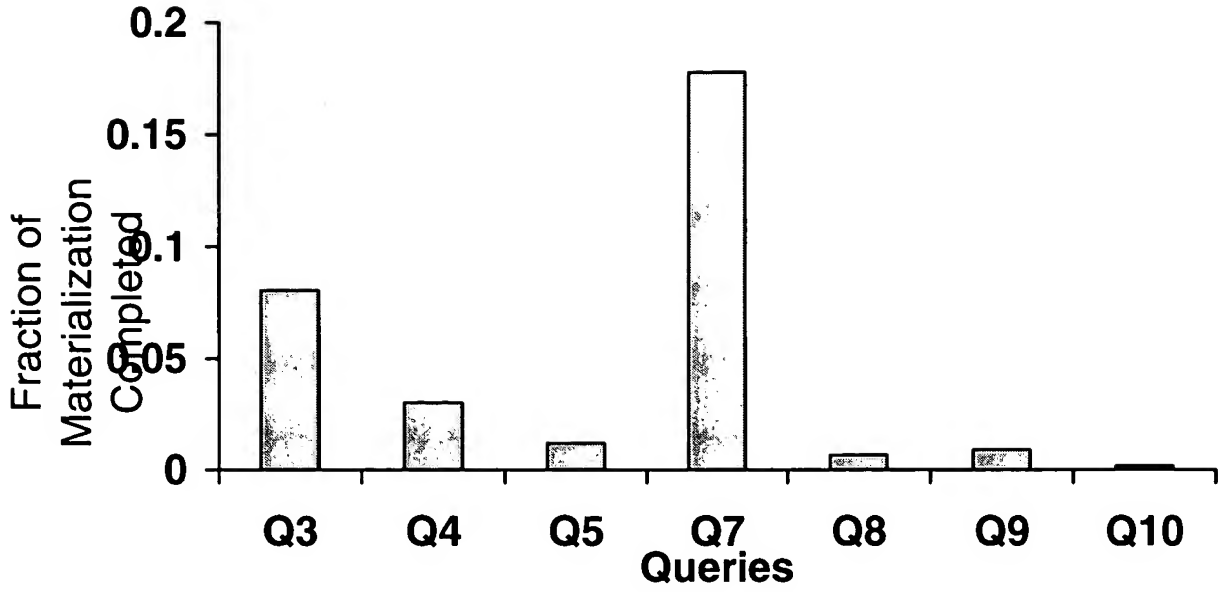

0.2
0.15
0.1
0.05
0
Fraction of Materialization Completed
Q3
Q4
Q5
Q7
Q8
Q9
Q10
Queries

FIG. 12

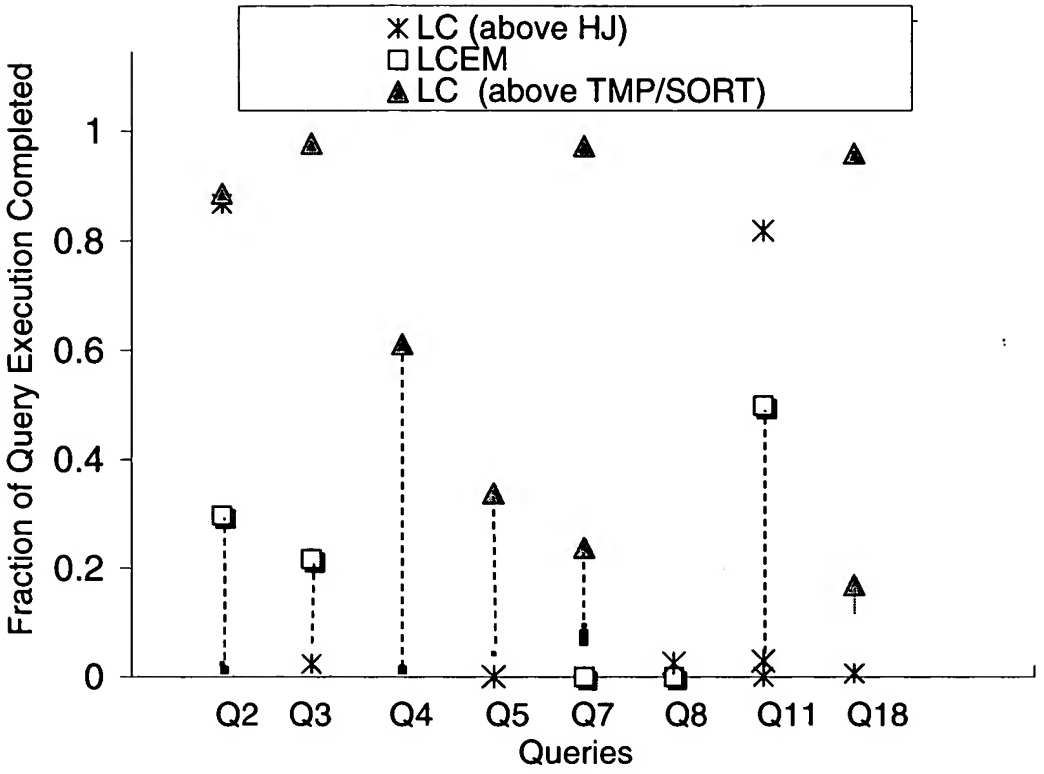

LC (above HJ)
LCEM
LC (above TMP/SORT)
Fraction of Query Execution Completed
1
0.8
0.6
0.4
0.2
0
Q2
Q3
Q4
Q5
Q7
Q8
Q11
Q18
Queries

FIG. 13

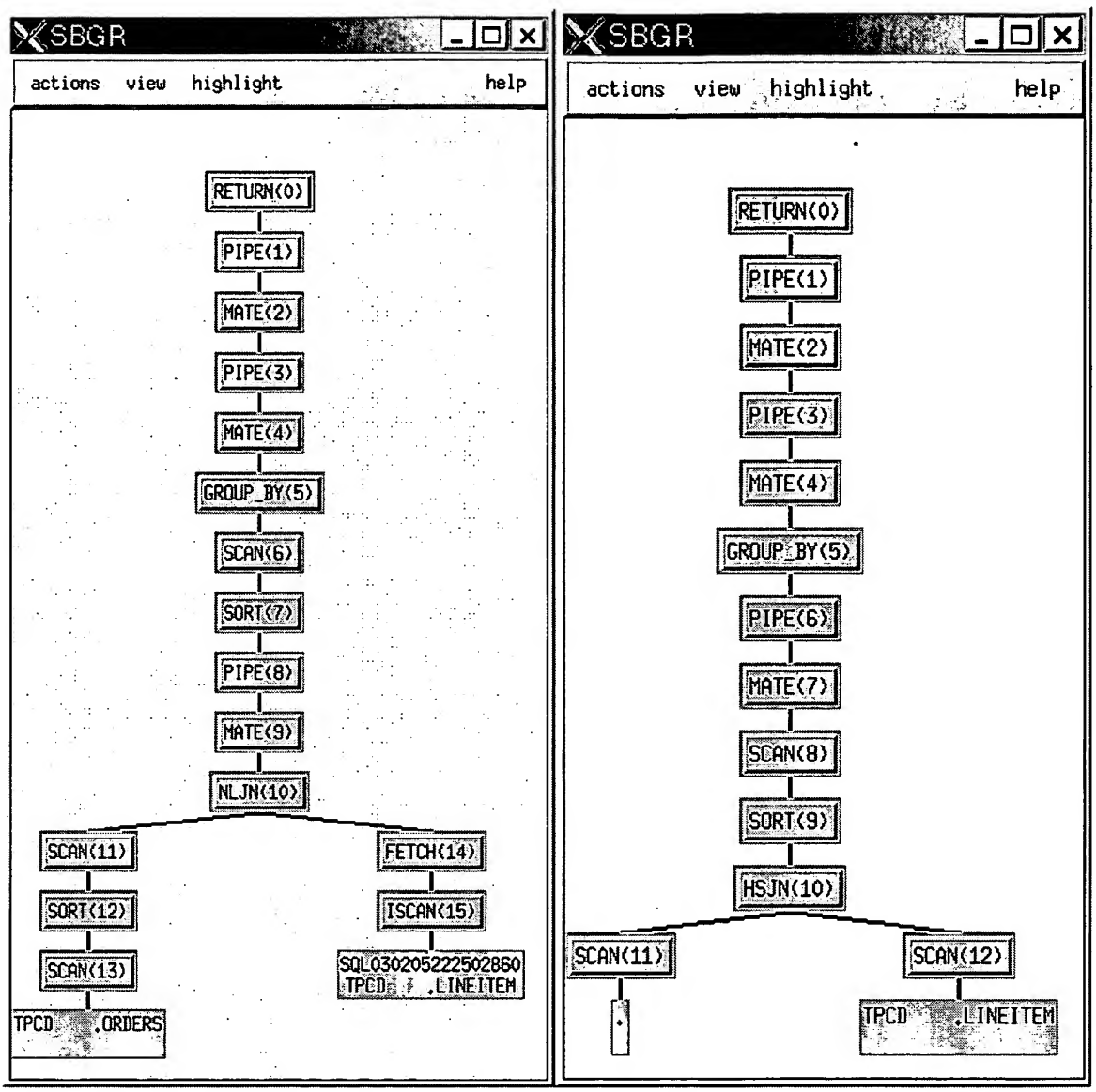

SBGR
actions view highlight help
RETURN(0)
PIPE(1)
MATE(2)
PIPE(3)
MATE(4)
GROUP_BY(5)
SCAN(6)
SORT(7)
PIPE(8)
MATE(9)
NLJN(10)
SCAN(11)
SORT(12)
SCAN(13)
TPCD .ORDERS
FETCH(14)
ISCAN(15)
SQL030205222502860
TPCD .LINEITEM
SBGR
actions view highlight help
RETURN(0)
PIPE(1)
MATE(2)
PIPE(3)
MATE(4)
GROUP_BY(5)
PIPE(6)
MATE(7)
SCAN(8)
SORT(9)
HSJN(10)
SCAN(11)
SCAN(12)
TPCD .LINEITEM

FIG. 14

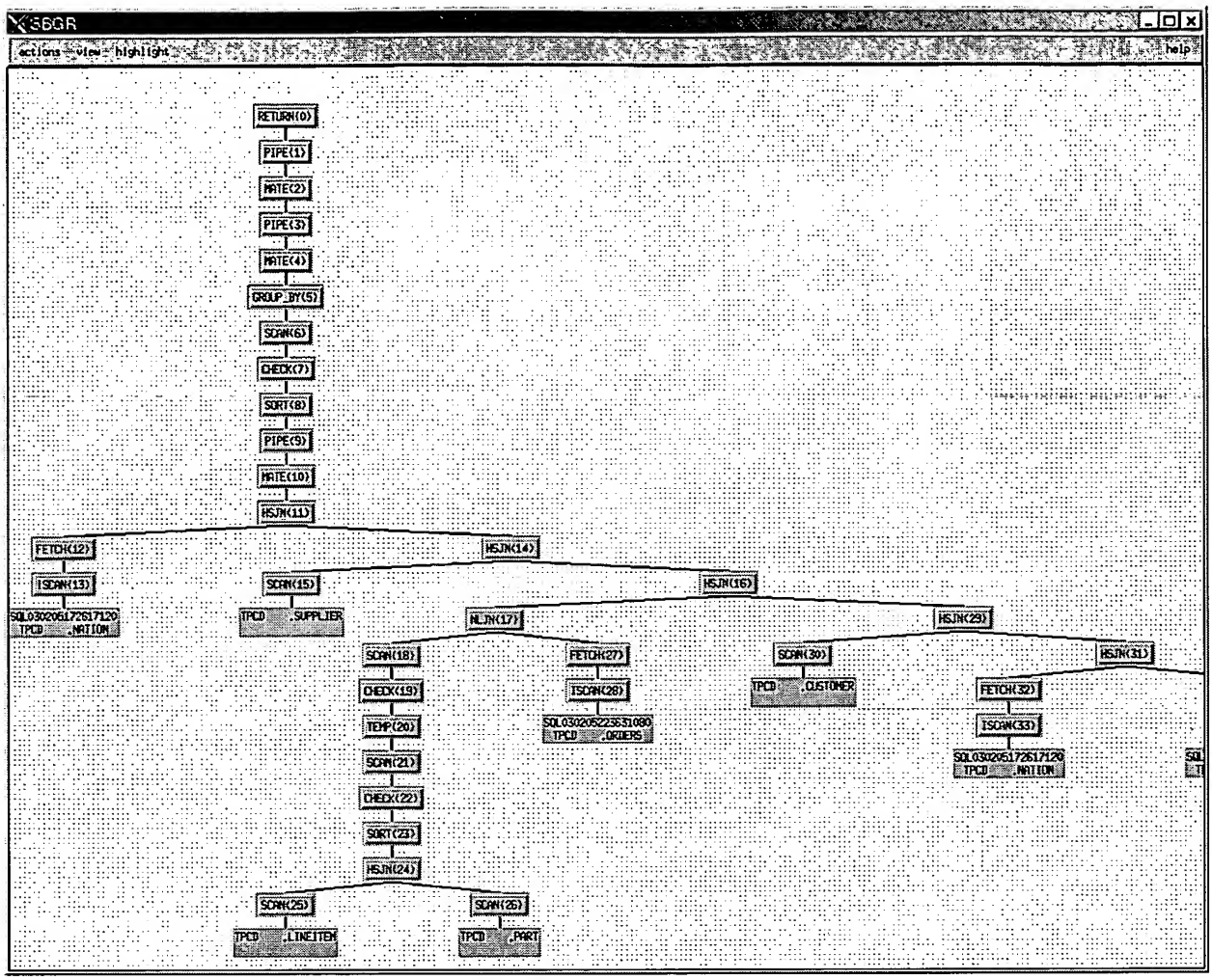

SBGR
actions
view
highlight
help
RETURN(0)
PIPE(1)
MATE(2)
PIPE(3)
MATE(4)
GROUP_BY(5)
SCAN(6)
CHECK(7)
SORT(8)
PIPE(9)
MATE(10)
HSJN(11)
FETCH(12)
ISCAN(13)
SQL0302051726171201
TPCD .NATION
HSJN(14)
SCAN(15)
TPCD .SUPPLIER
HSJN(16)
NLJN(17)
SCAN(18)
CHECK(19)
TEMP(20)
SCAN(21)
CHECK(22)
SORT(23)
HSJN(24)
SCAN(25)
TPCD .LINEITEM
SCAN(26)
TPCD .PART
FETCH(27)
ISCAN(28)
SQL0302052236310801
TPCD .ORDERS
HSJN(29)
SCAN(30)
TPCD .CUSTOMER
HSJN(31)
FETCH(32)
ISCAN(33)
SQL0302051726171201
TPCD .NATION

FIG. 15

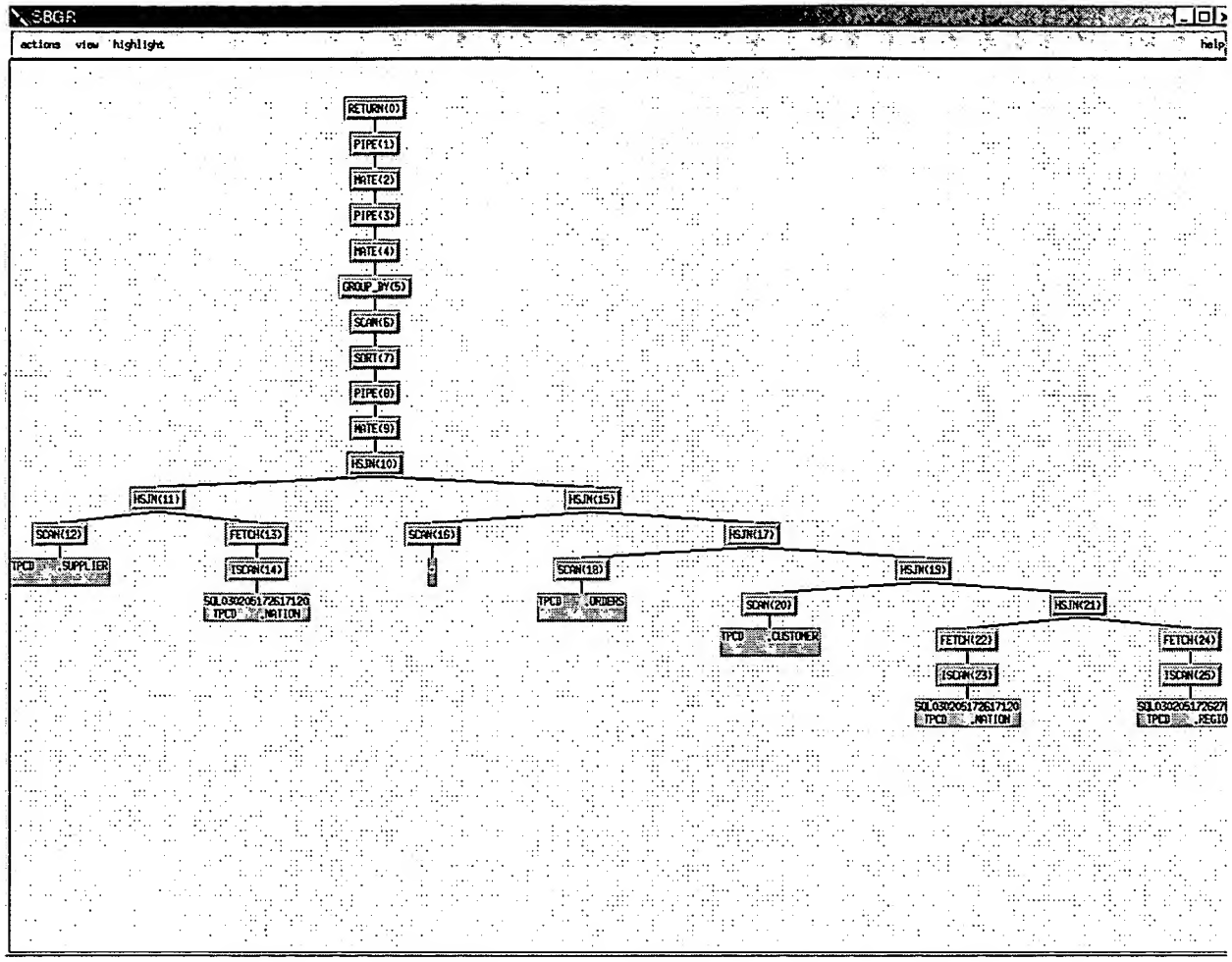

SBGR
actions view highlight help
RETURN(0)
PIPE(1)
MATE(2)
PIPE(3)
MATE(4)
GROUP_BY(5)
SCAN(6)
SORT(7)
PIPE(8)
MATE(9)
HSJN(10)
HSJN(11)
HSJN(15)
SCAN(12)
FETCH(13)
SCAN(16)
SCAN(18)
HSJN(17)
TPCD .SUPPLIER
TSCAN(14)
SQL0302051726171 20
TPCD .NATION
TPCD .ORDERS
SCAN(20)
HSJN(19)
TPCD .CUSTOMER
FETCH(22)
HSJN(21)
FETCH(24)
ISCAN(23)
ISCAN(25)
SQL0302051726171 20
TPCD .NATION
SQL0302051726271
TPCD .REGIO

FIG. 16

TABLE 1

Checkpoint Type	Placement	Risk	Opportunity
Lazy Check (LC)	CHECK Above materialization points	Very Low -- only context switching	Low, only at materialization points
Lazy Check with Eager Materialization (LCEM)	CHECK-Materialization pairs on outer of NLJN	Context Switching + materialization overhead	Materialization points and NLJN outers
Eager Check with Buffering (ECB)	BUFCHECK on outer of NLJN.	Lower than LCEM till SafeEagerThreshold, gradually increases afterward	Can reoptimize anytime during materialization
Eager Check without compensation (ECWC)	CHECK below materialization points	High – may throw away arbitrary amount of work during reoptimization	Anywhere below a materialization point
Eager Check with deferred compensation (ECDC)	CHECK and INSERT before reoptimization; anti-join afterward	High – may throw away arbitrary amount of work during reoptimization	Anywhere in the plan.